A Meta Model for Monitoring Requirements in Cloud Environnment

Rima Grati, Khouloud Boukadi and Hanêne Ben-Abdallah
Mir@cl Laboratory, University of Sfax, Faculty of Economics and Management of Sfax, BP 1088, Sfax 3018 Tunisia

Keywords: Monitoring of Web Service Composition, Cloud Environment, Profile UML.

Abstract: Cloud computing is a promising technology where the infrastructure, platform technology and software are remotely delivered as services over the Internet. Despite the increasing interest in cloud computing, several enterprises are still reticent to fully adopt it in their business because of several unresolved difficulties. Among others, the difficulty to provide guaranteed QoS for all kind of applications is one of the major barriers for the adoption of this computing paradigm by a wider range of enterprises. To overcome this difficulty, various service monitoring approaches were proposed as a means to detect potential violations of an agreed-upon service level. These approaches, however, tackled the technical aspects of monitoring without providing for a conceptual framework where the QoS monitoring requirements can be specified. This paper highlights the need for a design-level modelization of monitoring composed web services in the cloud. It then presents a meta-model for composite web services in the cloud, that provides for the specification of QoS requirements, web service level agreements, and monitoring QoS of composite web services in the cloud.

1 INTRODUCTION

Cloud computing is a rapidly spreading trend of computing where readily available computing resources are exposed as services. Depending on the user's needs, cloud computing offers services at different layers: Software (Software as a service: SaaS), Platform (Platform as a Service: PaaS) and Infrastructure (Infrastructure as a Service: IaaS). In recent years, SaaS implementations are in particular gaining popularity as a means to let both users manage typical day-to-day tasks and enterprises make money by arranging an ongoing software licensing agreement with different businesses.

Despite these advantages, given the complexity of the Cloud environment, service failures are quite likely and are the norm rather than the exception (Vishwanath and Nagappan, 2010). Service failures may result in QoS degradations at all layers. Hence, SaaS applications require QoS monitoring for two main reasons: on the one hand, to offer Cloud usage that is "acceptable" by various clients and, on the other hand, to spare Cloud providers penalties due to the violation of a QoS level agreed-upon in the Service-Level-Agreement (SLA).

To monitor the QoS of services, most works in the literature focussed on the technical side and neglected the conceptual side (cf., (Shao et al., 2010), (Cao et al., 2009) and (Clayman et al., 2010)). In addition, the proposed solutions require modification of the server and/or the client implementation code. In our previous work, we proposed a framework for QoS Monitoring and Detection of SLA Violations (QMoDeSV) (Grati et al., 2012). The framework QMoDeSV provides for monitoring composite services deployed on the Cloud with no intrusion on the client nor server implementation code.

In this paper, we complement QMoDeSV with a conceptual formalisation that offers three contributions to monitoring in the Cloud. As a first contribution, we propose a meta-model to specify QoS monitoring Requirements for Composite Services; this meta-model, called QoSReq4CS, can be used by Cloud customers to define their QoS requirements for their composite services deployed by a SaaS provider. In a second contribution, we propose a meta-model to facilitate the specification of SLA between a Cloud provider and its customers; this meta-model, called WSLA4CS, extends the WSLA specification (Ludwig et al., 2003) to take into consideration the cloud context. In addition, it formalizes the creation process of SLAs based on the QoS monitoring requirements. Our third

grati R., Boukadi K. and Ben-Abdallah H..
A Meta Model for Monitoring Requirements in Cloud Environnment.
DOI: 10.5220/0004442702230231
In *Proceedings of the 15th International Conference on Enterprise Information Systems* (ICEIS-2013), pages 223-231
ISBN: 978-989-8565-60-0

contribution is the meta-model MonitorQoS4CS which defines concepts related to the monitoring of composite web services in order to detect potential violations of the SLA contract.

The remainder of this paper is organized as follows: Section 2 overviews works related to monitoring Web services in the Cloud. Section 3 presents the meta-model for Composite Web Service in the cloud (CompositeWSinTheCloud) regrouping all three meta-models. Finally, Section 4 summarizes the presented work and highlights its future directions.

2 RELATED WORK

Most of the ongoing research efforts dealing with cloud monitoring, cf. (Shao et al., 2010), (Cao et al., 2009) and (Clayman et al., 2010), tackled the technical aspects and neglected the conceptual side of monitoring.

Shao et al. (Shao et al., 2010) propose a Runtime Model for Cloud Monitoring (RMCM). RMCM uses interceptors (as filters in Apache Tomcat and handlers in Axis) for service monitoring. It collects all Cloud layer performance parameters. In the SaaS layer, RMCM monitors applications while taking into account their required constraints and design models. To do so, it converts the constraints to a corresponding instrumented code and deploys the resulting code at the appropriate location of the monitored applications. In other words, RMCM modifies the source code of the applications being monitored. In this work, Shao et al. (Shao et al., 2010) do not propose any formalism for specifying the constraints, which represent certain QoS requirements to be monitored.

Cao et al. (Cao et al., 2009) propose a monitoring architecture for Cloud computing. In this not-yet-implemented architecture, cost is the only SLA monitoring requirements.

Clayman et al. (Clayman et al., 2010) propose the Lattice framework for Cloud service monitoring in the RESERVOIR EU project. Lattice is capable of monitoring physical resources, virtual machines and customized applications. This approach addresses some requirements and functionality of the service cloud environment such as QoS, elasticity, scalability, etc. Unlike our approach (Grati et al., 2012), the Lattice framework does not explain how the QoS requirements are specified.

Rak et al. (Rak et al., 2011) propose Cloud application monitoring using the mOSAIC approach. To benefit from mOSAIC, the application to be monitored must be first customized using mOSAIC API. Once customized, an application can be monitored by gathering low-level information used to perform manual or automatic load-balancing, increase/decrease the number of virtual machines, or calculate the total cost of the application execution. Besides being intrusive on the application code, the mOSAIC approach does not offer any formalism for specifying QoS requirements.

Boniface et al. (Boniface et al., 2010) propose a monitoring module that collects QoS parameters of Cloud Computing. They use a monitoring application component (AC) that must be first described and registered in the application repository. The AC collects QoS parameters at both the application and technical levels. This approach is complicated and hard to install due to the description and registration of AC. In addition, similar to the above approaches, this approach offers no means to describe the QoS requirements.

Patel et al. (Patel and anabahu, 2009) propose a mechanism for managing SLAs in a cloud computing environment using the Web Service Level Agreement (WSLA) framework. WSLA was developed for SLA monitoring and enforcement in a Service Oriented Architecture (SOA). This approach uses the third party support feature of WSLA to delegate monitoring and enforcement tasks to other entities in order to solve the trust issues.

Wenzel et al. (Wenzel et al., 2012) develop an approach to examine whether outsourcing of a business process in a cloud environment is possible while keeping all security and compliance requirements. The first pillar of their approach is a security risk analysis of the business process that are to be outsourced into a cloud. The second pillar of their approach is a compliance check that verifies the legal regulations constraints are still kept. In order to formulate such constraints, the meta model of the business process model is extended with a set of OCL expressions. The third pillar of their approach is the automated analysis of security properties. In their approach Wenzel et al. present how they specify the constraints, which represent certain security requirements but not in the context of monitoring QoS for composite web service deployed in the cloud to avoid SLA violations.

To the best of our knowledge, none of the examined approaches deals with the conceptual aspect for monitoring services in the Cloud. The exception is the work of (Patel and Ranabahu, 2009) who proposed an extension of WSLA to be adapted to the Cloud environment; the extension lacks however several concepts needed to link the SLA to

the QoS monitoring requirements of composite web services.

Our work complements existing technical monitoring frameworks/approaches by offering formalisms to model Composite Web Services in the Cloud in terms of Cloud customer requirements, SLA contracts, and necessary monitoring services to offer. Our proposed models will be used later for the generation of monitor capable of monitoring the functioning of a composite web service in the cloud according to the customer needs.

3 CompositeWSinTheCloud META MODEL

This section presents a meta-model for specifying Composite Web Service in the cloud (CompositeWSinTheCloud). This meta-model consists of the following three packages which describe the necessary concepts (see Figure 1):

- QoSReq4CS package: defines how a cloud customer can express their QoS requirements for a composite Web service deployed by a SaaS provider;
- WSLA4CS package: proposes an extension of the WSLA specification to take into consideration the cloud context. The aim of WSLA4CS is to create SLA approved between the customer and the monitoring service in the Cloud side;
- MonitorQoS4CS package: defines concepts related to the monitoring of composite web services in order to detect the validation or violation of their SLA contracts.

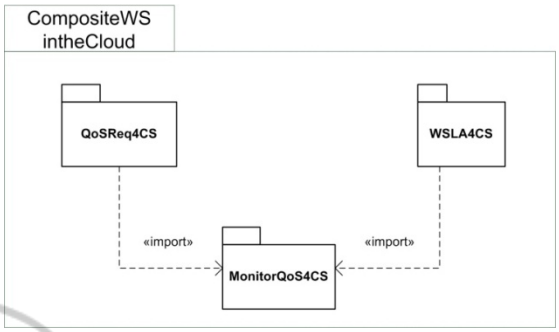

Figure 1: The CompositeWSinTheCloud meta-model.

3.1 The QoSReq4CS Package

The customer requests a particular composite Web service from a SaaS provider by submitting their QoS requirements (see Figure 2). QoS requirements describe: any QoS characteristics that can be measured such as performance, and QoS features considered as essential when the service is running such as reliability and availability. In the Web service field, there are several types of quality of service characteristics (Liu et al., 2004) (Canfora et al., 2005). In our work, we focus on those characteristics related to runtime issues rather than those related to the Web service development (such as reusability, testability, etc.).

The customer expresses their QoS requirements, for a composite Web service, thanks to QoS

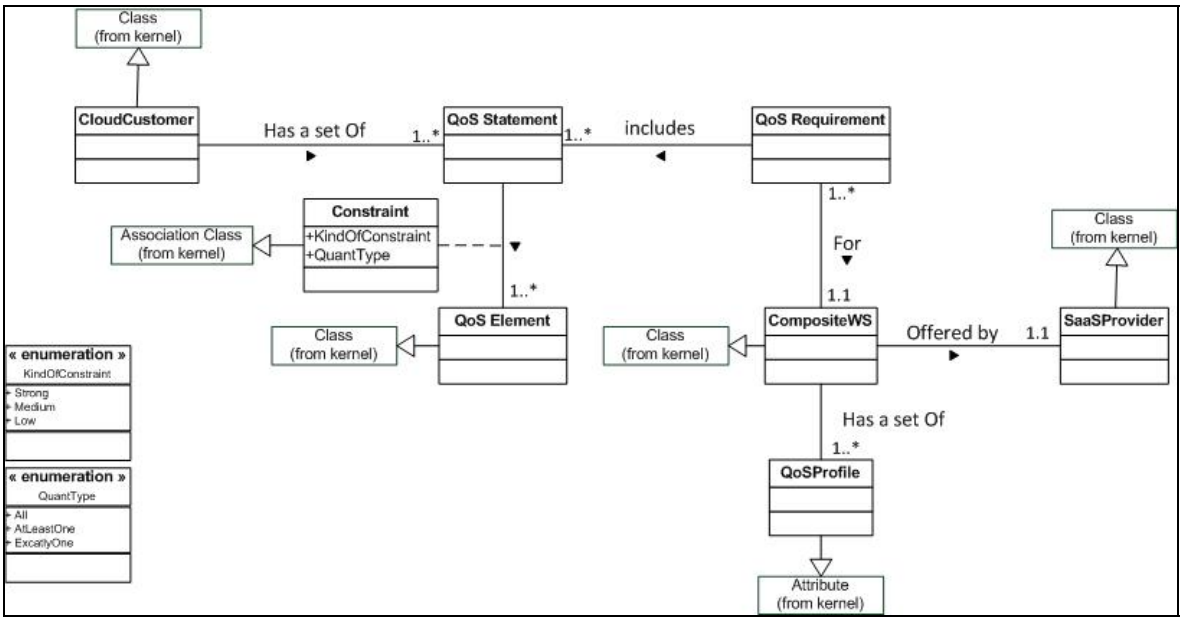

Figure 2: The QoSReq4CS package.

statements, which represent a set of constraints over QoS elements. In practical terms, each constraint imposed by a QoS statement must be ensured by the composite service once an agreement is made between the SaaS provider and the customer.

Within a QoS statement, the constraint over a QoS element is governed by a kind of constraint. In our approach, we adopt the notion of satisfiability proposed in (Mylopoulos et al., 1992) in which the QoS statements are achieved to a certain degree, rather than absolutely. Hence, we define three kinds of constraints: *strong*, *medium* and *low*. For example, a customer can state that the Web service throughput is qualified as a medium constraint, while the Web service reliability is a strong constraint over the reliability QoS element.

Furthermore, the combination of QoS elements within a specific QoS statement follows three quantifier types: *all*, *at least one* and *exactly one*.

We propose in the following a set of definitions that help the reader to understand the proposed concepts.

Definition1. QoS Requirement
A *QoS Requirement* is defined as a sequence of QoS Statements

$$QoS\_Req = <QoS\_St_1; \ldots ; QoS\_St_n>$$

where QoS_St_i is a QoS Statement.

Definition2. QoS Statement
A *QoS Statement* is defined as a couple

$$QoS\_St = (QuantType, QoSConstElts)$$

where

- QuantType ∈ {All, AtLeastOne, ExactlyOne} defines the participation of the QoS elements within QoS_St. The type is interpreted as follows:
  - ✓ All: QoS_St is defined by composing all the QoS elements,
  - ✓ AtLeastOne: QoS_St is defined by one or more of its QoS elements ,
  - ✓ ExactlyOne: QoS_St is defined by exactly one of its QoS elements.
- QoSConstElts is a set of constrained QoS elements (QoS_elt, KindConst) where
  - ✓ QoS_elt is a QoS element related to the Web service execution. Example of QoS elements can be response time, service cost, throughput, reliability, etc.

KindConst ∈ {*Strong, Medium, Low*} defines the constraint regarding a QoS element.

Definition 3. Composite Web service
A composite Web service is defined as a structure

$$CompositeWS = <Name, Desc, CS, CP, QoSProf>$$

where

- Name: is the name of the composite Web service, used as its unique identifier;
- Desc: is the description of the composite Web service. It summarizes what the service offers;
- CS: is the set of its component services,
- CP: is the set of the composition patterns to indicate the control flow between the component services. These patterns are Sequence, Parallel, Synchronization, Exclusive, Simple merge, Conditional, Synchronizing merge, Multi merge, Loop and Deferred choice pattern. For further information about these patterns the reader can refer to (WorkflowPatterns, 2013).
- QoSProf: a set of QoS profils (QoS profs). Each QoSProf is a template for a particular quality of service computed based on the used composition pattern as well as the IT resources responsible for executing the service. For example the provider can offer a service with set of QoS profils: the first QoS profil consists of response time 5ns and costs 10 euro, the second QoS profil is response time 10ns and cost 7 euro.

3.1.1 SLA between Cloud Customer and SaaS Provider

In our work, we assume that a request for a particular Web service is sent from the cloud customer to a SaaS provider's application layer with QoS requirements (Figure 3). Thereafter, the provider looks for the QoS profile that best fits the customer's request and proposes an SLA template. In case of approval (i.e. the customer agrees on the SLA template offering the desired QoS level), the customer sets the contract validity period and proposes the SLA to the provider. The provider may reject the proposed SLA, otherwise it is accepted and the contract is established.

The SaaS provider either rents resources from IaaS providers, or proposes its own resources in order to lease the composite Web service to customers. In our work, we assume that a SaaS provider has its own resources and hence it is in charge of the required resource allocation. In addition, it is responsible for dispatching VM images to run on the physical resources and to create instances for executing services.

The scenario depicted in Figure 3 can be achieved thanks to an SLA negotiation protocol which includes the involved parties, the service description as well as the QoS that should be met by

the provider. Flexible and reliable management of SLA agreements is of paramount importance for both cloud customer and provider. Indeed, several authors in the literature have demonstrated that SLA monitoring is a prominent issue in the current cloud computing context (Alhamad et al., 2010) and (Patel and Ranabahu, 2009). For example, in (Patel and Ranabahu, 2009) the authors emphasize that, due to the dynamic nature of cloud computing, an independent tool for monitoring/validating performance of application is one of the facilities which are most needed in the cloud.

The SLA monitoring facility enables both customer and provider to understand if any failure or quality of service degradation is caused by the cloud provider, the network infrastructure, or even the design of the service itself (Emeakaroha et al., 2012).

There are two main specifications for describing SLA for Web services: Web service Agreement (WS-Agreement) (Andrieux et al., 2004) from the Open Grid forum, and Web Service Level Agreement language and Framework (WSLA) from IBM (Ludwig et al., 2003). However, these proposals are not suitable for a cloud computing environment because the nature and type of IT resources provided and delivered are different (Patel and Ranabahu, 2009). In addition, a composite Web service runs on the cloud provider infrastructure which is shared and virtualized. However, current monitoring infrastructures rely either on Grids or service oriented infrastructures which are not compatible with the cloud context.

On the other hand, most of the proposed monitoring infrastructures require modification of either the server or the client implementation code. However, to provide for independence of any Cloud provider/environment, monitoring should be performed without modifying the implementation of the deployed services. Moreover, the cloud environment is dynamic and the resource usage changes frequently. So, when trying to enforce an SLA, one should take into account this dynamicity by proposing a set of functionalities that track the evolution of services.

Hence, a new SLA model adapted for the cloud context is still needed (Patel, 2009). In our work, we propose to enhance the WSLA in order the meet the cloud computing context. Our assumption is that the SLA monitoring facility is provided by the provider to the customer. To do so, we propose a set of services that run in parallel with the composite Web service instance in order to detect potential SLA violations. The extension of the WSLA is detailed in the next section.

3.2 The WSLA4CS Package and SLA Monitoring

The WSLA specification consists of a flexible and extensible language based on XML Schema and a runtime architecture comprising several SLA monitoring services, which may be outsourced to third parties to ensure a maximum objectivity. For

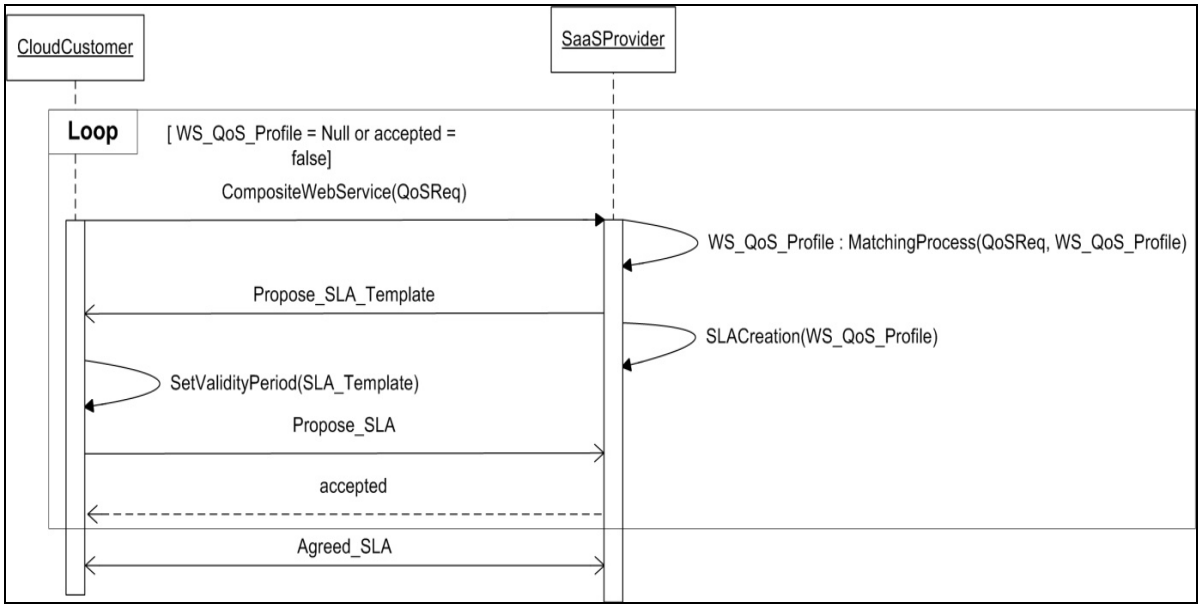

Figure 3: SLA agreement between Cloud Customer and SaaS Provider.

space limitation, we just present the definition of WSLA without detailing its various concepts. For more details the reader can refer to (Ludwig et al., 2003).

Figure 4 depicts the new WSLA4CS for specifying and monitoring SLAs for Web services deployed in a Cloud; the concepts we added to WSLA are highlighted.

The WSLA4CS is designed to achieve an agreement between a SaaS provider and a Cloud Customer concerning a cloud service definition. For this, the following four monitoring services are added to the initial specification:

- The QoS Calculator service *(QoSCalculatorService)* computes QoS metrics for the composite Web services. In its computations, it uses the values of the constituent services and the composition pattern defined in the monitored service. The overall Web service QoS is derived based on the values collected for each constituent service and the composition pattern. For this derivation, the calculation formulas proposed in (San-Yih, et al., 2004) and (Jaeger et al., 2004) can be easily adapted.
- The host monitor service *(HostMonitorService)* is responsible for measuring the runtime parameters of cloud provider resources by retrieving resource metrics directly from the provider managed resources based on two fundamental principles of the cloud: the pay as you go and the elasticity feature of the cloud. It maintains information on the current system configuration and runtime information about the metrics which are part of the SLA. It can be configured to access different virtual hosts at the same time to collect locally monitored values. The particularity of the host monitor service is that it can take into account two types of metrics including: resource metrics and JVM metrics. JVM metrics are considered by the host monitor service because of the nature of the monitored service (i.e. composite Web service running thanks to JVM applications).
- The Lo2Hi QoS Convertor service *(Lo2HiQoSConvertorService*) interacts with two components: the host monitor which monitors the resources, and the QoS Calculator which calculates the global obtained metrics. Resources are monitored by the host monitor using arbitrary monitoring tools such as Gmond from Ganglia project (Massie et al, 2004). Based on the predefined mapping rules stored in a database, monitored metrics are periodically mapped to the SLA parameters.
- The SLA Violation Detector service*(SLAViolationDetectorService)* accesses the

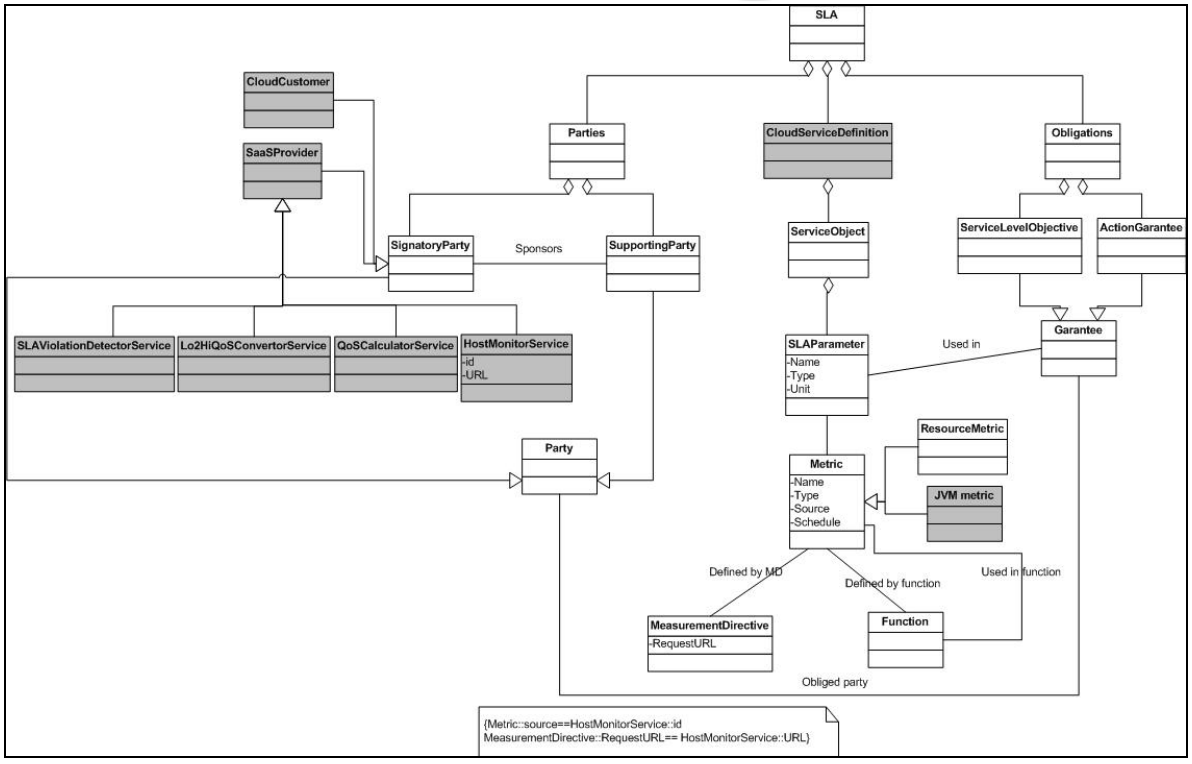

Figure 4: The WSLA4CS package.

mapped metrics repository to get the mapped SLA parameters. These parameters are compared with the calculated values obtained from the QoS Calculator. In the case of a violation (none respect of SLA), it dispatches notification messages to the customer/provider to alert about the violation.

Figure 5 shows the scenario when the client invokes a service. A cloud customer sends a request to the Seas provider with an approved SLA. The Seas provider allocates the necessary resources (infrastructure layer) and the platform adequate to meet the request of the customer. Once the resources are allocated, the provider begins the phase of monitoring based on all the services belonging to QMoDeSV framework. For further information about these services, we refer the reader to (Grati et al, 2012).

3.3 The MonitorQoS4CS package

The Monitoring QoS for Cloud Service package covers all concepts of WSLA4CS and QoSReq4CS.

This package defines concepts related to the monitoring of the composite service to detect potential violations of specified SLA. It presents also the different modules responsible for the monitoring service. In our work, we consider that the monitoring service is offered by the provider to the customer. This assumption is taken after a certain level of trust between the two parties is established.

4 CONCLUSIONS

In this paper, we discuss our approach for monitoring quality of services (QoS) guarantees given to the customers in Saas Cloud environments, because this compliance of the guarantees are not easily to maintain. Our goal is to monitor the requirements that arise from the QoS guarantees that the consumer has given.

For this purpose we presented a meta-model for monitoring Composite Web Service in the Cloud (CompositeWSinTheCloud). This meta-model consists of three packages that describe concepts pertinent to: expressing customer's QoS monitoring requirements for a composite service deployed by a SaaS provider (QoSReq4CS); expressing SLA specifications while taking into account the Cloud context (WSLA4CS); and defining concepts related to the monitoring of the composite service to detect potential violations of specified SLA (MonitorQoS4CS).

Currently we are working on the evaluation of the expressive power of the proposed meta-model through a real case study. We choose an example of B2B scenario consisting of a real estate company,

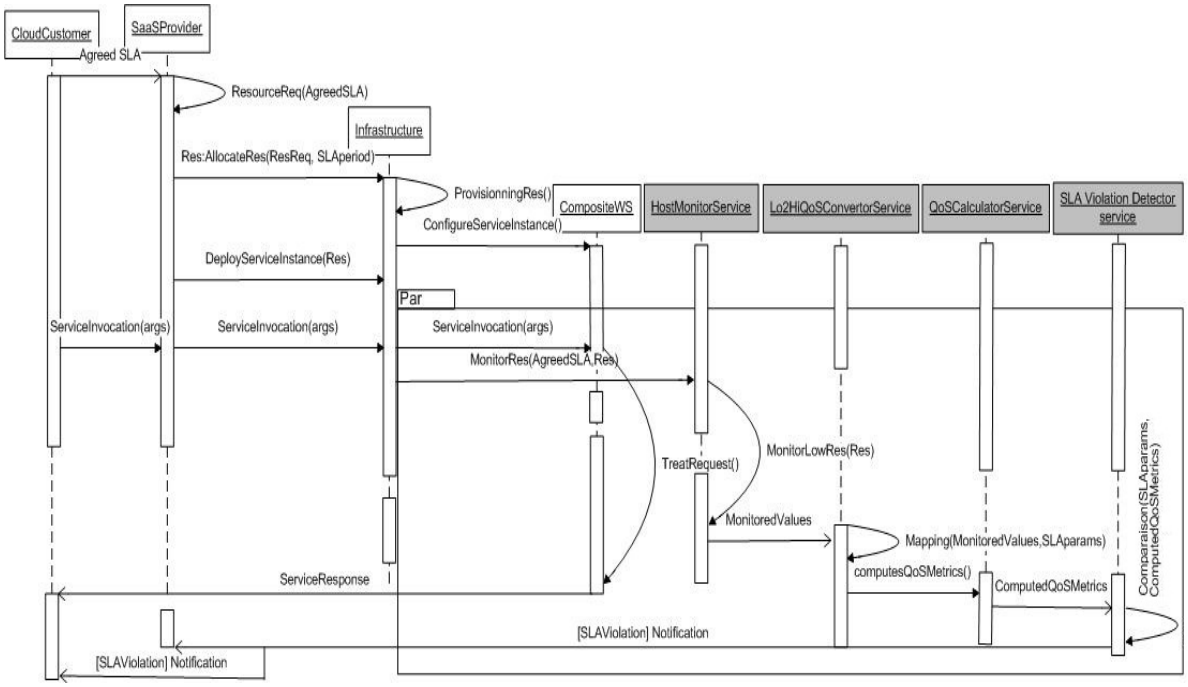

Figure 5: SLA monitoring for composite Web service.

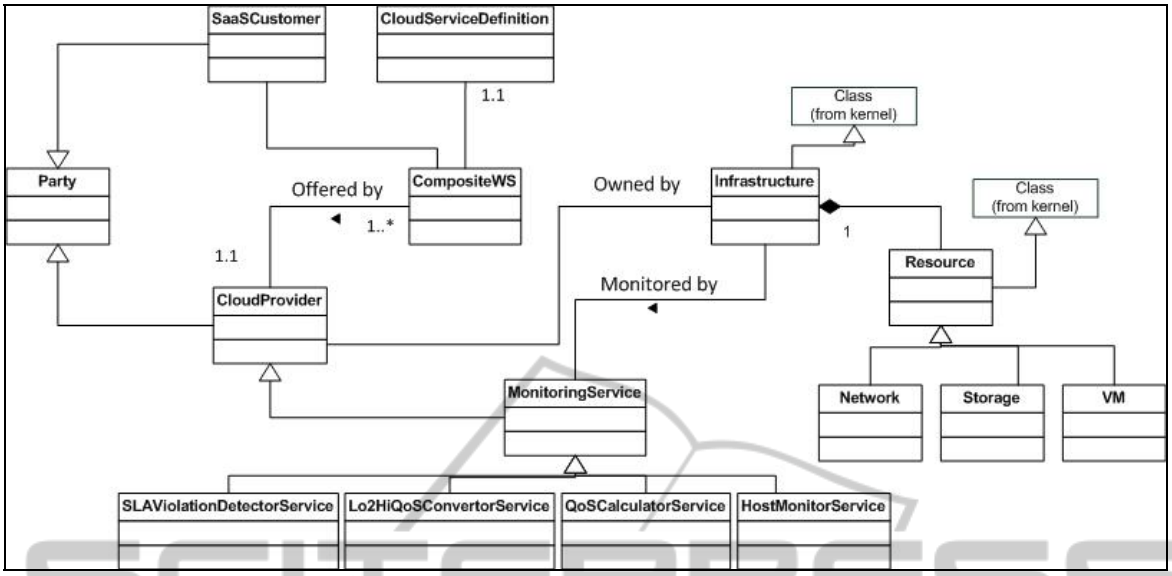

Figure 6: The MonitorQoS4CS package.

wishing to have a credit agreement to decide to build a building, addresses to a financial institution.

REFERENCES

Vishwanath, K. V. and N. Nagappan. Characterizing cloud computing hardware reliability. In Symposium *on Cloud Computing* (SOCC), 2010

Grati, R., Boukadi, K. and Ben-Abdallah, H. "A QoS Monitoring Framework for Composite Web services in the Cloud", In the *6th International Conference on Advanced Engineering Computing and Applications in Sciences* 2012 (Advcomp'12). pp. 65-70

Ludwig, H., Keller, A., Dan, A., King, R., Franck, R, Web service level agreement (WSLA) language specification. *IBM Corporation* (2003)

Shao, J., Wei, H., Wang, Q. and H. Mei, "A Runtime Model Based Monitoring Approach for Cloud," in Proceedings of 2010 IEEE 3rd International Conference on Cloud Computing (CLOUD 2010), I. C. Society, Ed. Miami, Florida: *IEEE Computer Society, 2010*, pp. 313–320.

Cao, B.-Q. , Li, B and Xia, Q.-M "A Service-Oriented Qos-Assured and Multi-Agent Cloud Computing Architecture," in Proceedings of the *1st International Conference on Cloud Computing* (CloudCom'09). Berlin, Heidelberg: *Springer- Verlag, 2009, pp*. 644–649.

Clayman, S., Galis, A., C. Chapman, M. L.R, L. M. Vaquero, K. Nagin, B. Rochwerger, and G. Toffetti. "Monitoring future internet service clouds" In Towards the Future Internet - *A European Research Perspective book*, April 2010.

Rak, M., Venticinque, S., Mandhr, T. a., Echevarria, G. and Esnal, G. "Cloud application monitoring: The mosaic approach". In Cloud Computing Technology and Science (CloudCom), 2011 IEEE *Third International Conference on, pages 758 –763, 29* 2011-dec. 1 2011.

Boniface, M., Nasser, B., Papay, J., Phillips, S. C., A. Servin, Yang, X., Z. Zlatev, S. V.Gogouvitis, G. Katsaros, K. Konstanteli, G. Kousiouris, A. Menychtas, and D. Kyriazis, "Platformas- a-Service Architecture for Real-Time Quality of Service Management in Clouds," in Proceedings of the 2010 *Fifth International Conference on Internet and Web Applications and Services (ICIW '10). Washington, DC, USA: IEEE Computer Society*, 2010, pp. 155–160

Patel Pankesh, Ajith Ranabahu, "Service level agreement in cloud computing, *UKPEW, 2009.*

Liu, Y. A. H. H. Ngu, and L. Zeng. Qos computation and policing in dynamic web service selection. In WWW (Alternate Track Papers & Posters), pages 66–73. *ACM, 2004.*

Wenzel S., C. Wessel, T. Humberg, and J. urjens. Securing processes for outsourcing into the cloud. In *CLOSER, Porto, Portugal*, 2012.

Canfora, G. ., M. D. Penta, R. Esposito, and M. L. Villani. An approach for qos-aware service composition based on genetic algorithms. In *GECCO, pages 1069–1075. ACM, 2005.*

Mylopoulos, J., Lawrence Chung, and Brian Nixon. Representing and Using Non-Functional Requirements: A Process-Oriented Approach. *IEEE Transaction of Software Engineering*, 18(6):483-497, June 1992.

WorkflowPatterns. [cited 2103 April 2013]. Available from: http://www.workflowpatterns.com/

Alhamad, M. T.Dillon, E.Chang, (2010), Conceptual SLA Framework for Cloud Computing", *4th IEEE International Conference on Digital Ecosystems and Technologies* (IEEE DEST 2010) 2010 IEEE

Emeakaroha, E.; Netto, M. A. S.; Calheiros, R.; Buyya, R.; De rose, C. A. F.; Towards Autonomic Detection of SLA Violations in Cloud Infrastructures. Future Generation of Computer Systems: *the International Journal of Grid Computing: Theory, Methods and Applications* (FGCS), 2012, Elsevier

Andrieux, A., Czajkowski, K., Dan, A., Keahey, K., Ludwig, H., Pruyne, J., Rofrano, J., Tuecke, S., Xu, M, Web services agreement specification (WS-Agreement*). In Global Grid Forum*. (2004)

San-Yih, H. Wang H, S.Jaideep , and P. Raymond. "A probabilistic QoS Model and computation Framework for Web Services based workflow" In *Proc of ER2004, pages 596-609, Sanghai*, November 2004. pp. 254-260

Jaeger Michael C., Gregor Rojec-Goldmann, and Gero Muhl. "QoS Aggregation for Web Service Composition using Workflow Patterns" EDOC '04 *Proceedings of the Enterprise Distributed Object Computing Conference, Eighth IEEE International*. pp. 52-59

Massie, M. L., B. N. Chun, and D. E. Culler, "The ganglia distributed monitoring system: Design, implementation and experience," *Parallel computing, vol. 30, pp. 200, 206*